



Lab Sheet: Python Fundamentals

This lab sheet covers the section on Python Fundamentals Classes from the course on *Industrial Programming* (F21SC): <http://www.macs.hw.ac.uk/~hwloidl/Courses/F21SC#slides>. These exercises should be implemented in Python 3.8. The tasks can be performed on your local machine (using either Linux or Windows) or on the Linux lab (EM 2.50) machines in the department (accessible through `x2go`).

Most exercises are available from the `gitlab-student.macs.hw.ac.uk` server, and these are marked as . Use this HOWTO do exercises with the gitlab server for an introduction how to do these exercises and how to use the unit testing in the continuous integration backend to get immediate feedback on your solution. Try this hello world exercise , alongside watching the HOWTO, to try this yourself.

Python Fundamentals (Week 8)


Get started with this exercise (**Gitlab** version of the lab exercise ):

(Sums1) Implement a function computing the *sum over a range of values*:

Given an integer value n as input, compute the sum of all values up to n : $1 + 2 + \dots + n$

(Sums2) Implement a function computing the *sum over the elements of a list*:

Given a list of integer values, compute the sum of all list elements: *Example*: if the input list is: `[2,3,5,7]` compute $2 + 3 + 5 + 7$

Then, pick up this file with list-based Python exercises (from Google's Python classes) and complete exercises (D) and (E)¹. In particular (**Gitlab** version of the lab exercise ):

(D) Implement a `remove_adjacent` function that collapses adjacent, equal entries in a list to just 1 entry.

Example: `remove_adjacent([1, 2, 2, 3])` \rightarrow `[1, 2, 3]`

(E) Given two lists sorted in increasing order, return a merged list of all the elements in sorted order.

Example: `linear_merge(['aa', 'xx', 'zz'], ['bb', 'cc'])` \rightarrow `['aa', 'bb', 'cc', 'xx', 'zz']`

Then, implement these exercises (**Gitlab** version of the lab exercise ):

(Factor1) Implement an integer factorisation function in Python: **Given**: an integer number n

Find: a list of all prime numbers dividing n , such that the product of the list is n

Example: `factor(120)` \rightarrow `[2, 2, 2, 3, 5]`

Hint: you can build on the function discussed in class for testing whether a number is a prime number

Testing: you can test the result by typing e.g. `factor 120` on the command line


(Factor2) Extend the previous implementation to record the result as a list of pairs, consisting of the prime number and the number of occurrences in the list above


Example: `factor(120)` \rightarrow `[(2, 3), (3, 1), (5, 1)]` for 3 occurrences of the prime number 2 and 1 of occurrence of 3 and 5 each

Hint: extend the previous example with code similar to `remove_adjacent` above

(Factor3) Extend the previous implementation by only adding the value in cases where the number of occurrences is 1. Note that in Python a list can have entries of mixed types.

Example: `factor(120)` \rightarrow `[(2, 3), 3, 5]` for 3 occurrences of the prime number 2 and 1 of occurrence of 3 and 5 each

(GCD) Implement Euclid's greatest common divisor algorithm as a function with two `ints`. (**Gitlab** )

(MatMult) Implement matrix multiplication as a function taking two 2-dimensional arrays (of any size) as arguments. (**Gitlab** ) Use the `numpy` library, and check the sample usage for this library at the end of this slideset.

¹©Copyright 2010 Google Inc. Licensed under the Apache License, Version 2.0